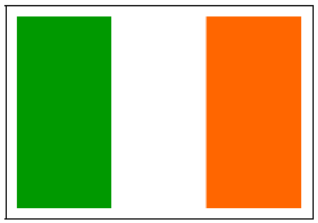
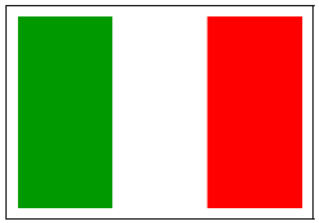
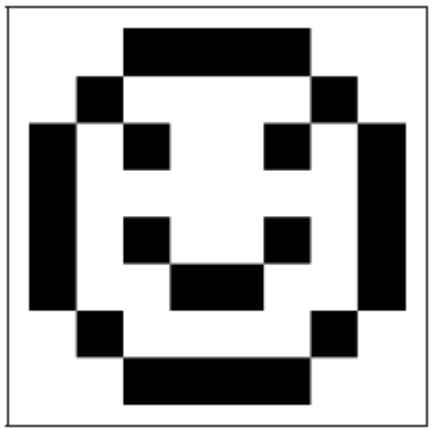
# 情報科学

- 第1章 数の計算と関数
  - ファイルを読み込むファイル・定数関数・局所変数
- 第2章 配列による画像の表示
- 第3章 条件分岐と繰り返し
  - 3.1 条件分岐
  - 3.4 繰り返しによる画像の作成



# 復習：関数定義と ファイルからの読み込み

bmi.rb

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight )
    weight / ( height / 100.0) ** 2
end
```

コメント

保存済ファイルを Emacsで**開く**方法: bmi.rbを「E」アイコン上に ドラッグ&ドロップ

ファイルを保存したディレクトリでirbを起動

```
irb(main):003:0> load("./bmi.rb")
=> true

irb(main):005:0> bmi(188.0, 104.0)
=> 29.4250792213671
```

# ファイルを読み込むファイル

```
load ("./bmi.rb")
load ("./yardpound.rb")

def bmi_yp(f,i,p,o)
    bmi(feet_to_cm(f,i), pound_to_kg(p,o))
end
```

bmi_yp.rb

```
def feet_to_cm(feet,inch)
  …
end
def pound_to_kg(pound,ounce)
  …
end
```

yardpound.rb

# 定数関数

bmi.rb

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight )
   weight / ( height/100.0) ** 2
end
def k_height()
   188.0
end
def k_weight()
   104.0
end
```

ファイルの中ではk_height = 188.0とは書けない

```
irb(main):004:0> load("./bmi.rb")
=> true
irb(main):005:0> k_weight()
=> 104.0
irb(main):006:0> bmi(k_height(), k_weight())
=> 29.4250792213671
```

# 局所変数

```
include(Math)
def heron(a,b,c)
    s = 0.5*(a+b+c)
    sqrt(s * (s-a) * (s-b) * (s-c))
end
```

heron.rb

数学関数を使う場合は
ファイルの先頭に書く

「s=0.5(a+b+c)とおく」

最後の式が関数の答え

$$s = \frac{a+b+c}{2}$$

$$\sqrt{s(s-a)(s-b)(s-c)}$$

# 字下げ

```
include(Math)
def heron(a,b,c)
    s = 0.5*(a+b+c)
    sqrt(s * (s-a) * (s-b) * (s-c))
end
```

```
def fibr(k)
    if k==0 || k==1
        1
    else
        fibr(k-1) + fibr(k-2)
    end
end

def fibl(k)
    f=1
    p1=1
    for i in 2..k
        p2 = p1
        p1 = f
        f  = p1 + p2
    end
    f
end

def fibl6(k)
    f=1
    p1=1
    for i in 2..k
        p2 = p1
        p1 = f
        f  = (p1 + p2) % 1000000
    end
    f
end
```

あり

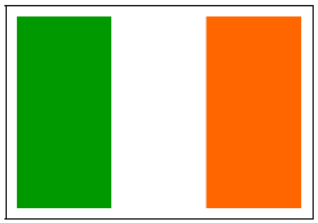
```
def fibr(k)
if k==0 || k==1
1
else
fibr(k-1) + fibr(k-2)
end
end

def fibl(k)
f=1
p1=1
for i in 2..k
p2 = p1
p1 = f
f  = p1 + p2
end
f
end

def fibl6(k)
f=1
p1=1
for i in 2..k
p2 = p1
p1 = f
f  = (p1 + p2) % 1000000
end
f
end
```

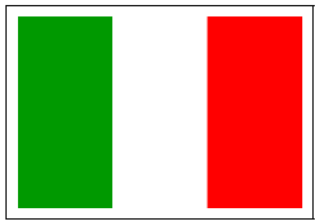
なし

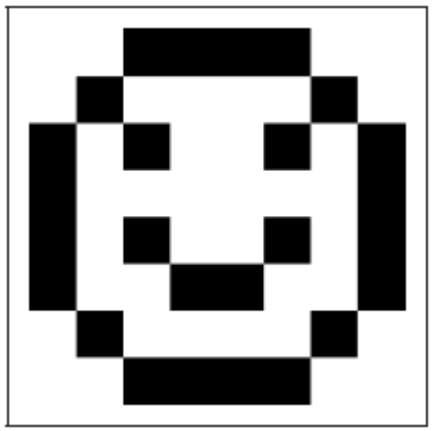
字下げ：行の先頭につける空白のこと

- 関数定義の本体は字下げしよう
  - 「どこまでか」が見やすくなる
  - 他にも「範囲」のあるものは同様 (for, ifなど)
- EmacsはTabを押すと自動的に字下げされる

# 2 画像の表現



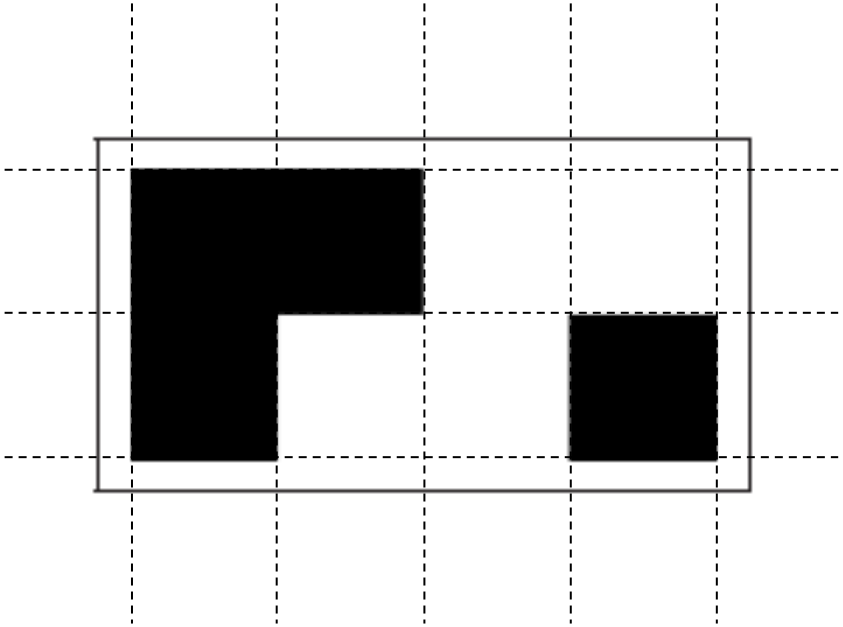




# データの表現 — 画像の表現 (p.26)

```
irb(main):002:0> コントロールD
cm12345$ isrb
>> a = [[0,0,1,1],
?>      [0,1,1,0]]
=> [[0 , 0, 1, 1], [0, 1, 1, 0]]

>> show(a)
=> nil
```

isrb のプロンプト



irb(main):002:0> コントロールD

cm12345$ isrb

>> a = [[0,0,1,1],

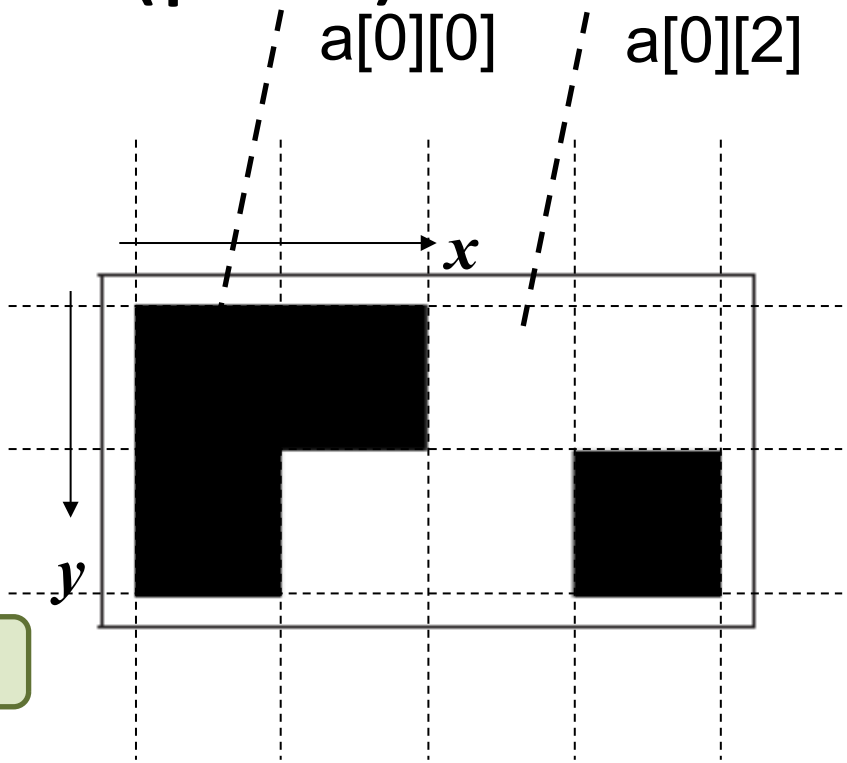
?>  [0,1,1,0]]

=> [[0 , 0, 1, 1], [0, 1, 1, 0]]

>> show(a)

=> nil

# 画像の操作 (p.27)




>> a[0][0]

=> 0

>> a[0][2]

=> 1

明度を変更

>> a[1][2]=0.5

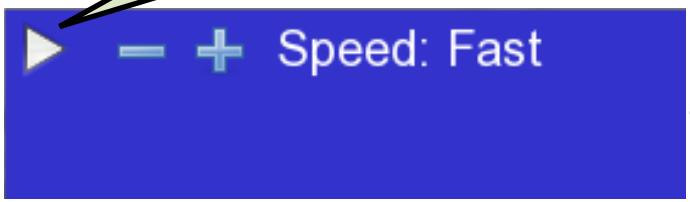
=> 0.5

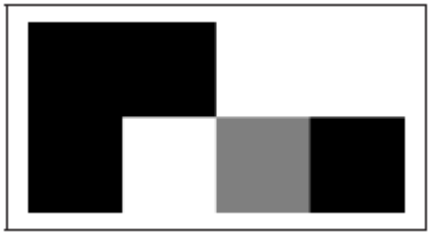
再表示

>> show(a)

=> nil

ここをクリック



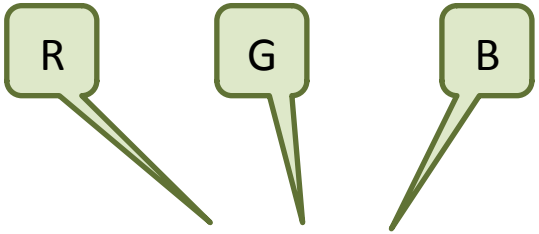
# クイズ: 次の結果は何？

```
irb(main):001:0> a = [[3,1,4,1,5,9],[2,6,5,3,5,8]]
=> [[3,1,4,1,5,9],[2,6,5,3,5,8]]
irb(main):003:0> a[1][3]
=> 3
irb(main):004:0> a[0][0]=a[1][1]
=> 6
irb(main):004:0> a[0][0]+a[1][0]
=> 8
```

irbを**使わずに**

「ターミナル」のウィンドウを新たに開き、
**cd Downloads**
**ruby c.rb 答の数**

# 2.3 カラー画像の表現 (p.28)

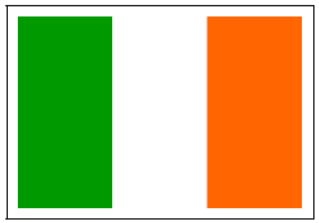
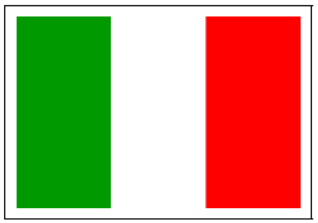



```
>> d=[[[0 ,0 ,0] ,[0 ,1 ,0] ,[0 ,0 ,1]] ,
?>      [[1 ,0 ,0] ,[1 ,1 ,0] ,[1 ,0 ,1]]]
=> [[[0 , 0, 0], [0, 1, 0], [0, 0, 1]] , [[1 , 0, 0],
[1, 1, 0], [1, 0, 1]]]
>> show (d)
=> nil
```

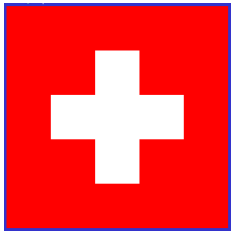
# 練習 (画像)

練習2.4 (国旗, p.31)

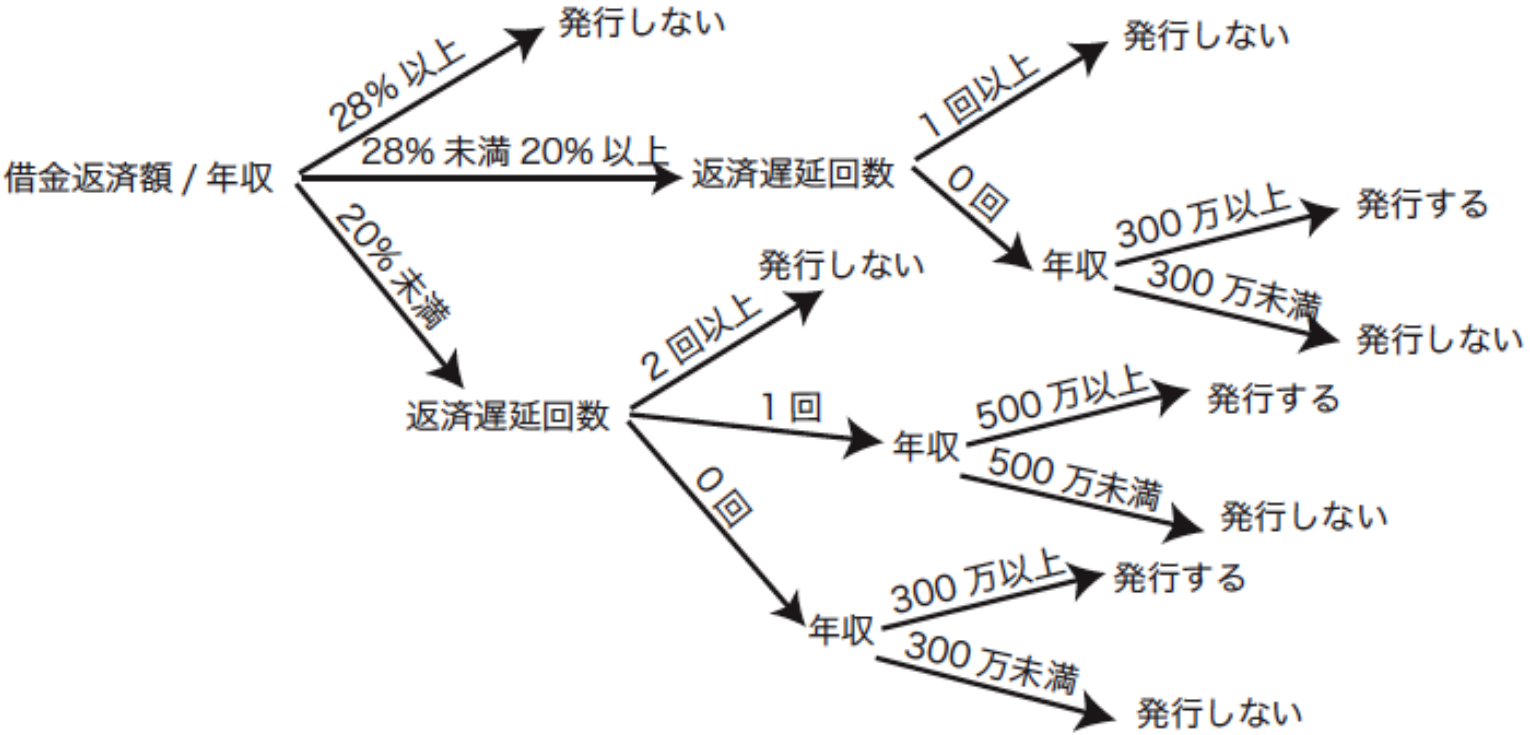
※ flag.rbというファイルにflag()という関数を定義するとよい


```
def flag()
  r=[1,0,0]
  w=[1,1,1]
  [[r,r,r,r,r],
   [r,r,w,r,r],
   [r,w,w,w,r],
   [r,r,w,r,r],
   [r,r,r,r,r]]
end
```



# 3.1 条件分岐 (p.33)


借金返済額 / 年収
28% 以上
発行しない
28% 未満 20% 以上
返済遅延回数
1 回以上
発行しない
0 回
年収
300 万以上
発行する
300 万未満
発行しない
20% 未満
返済遅延回数
2 回以上
発行しない
1 回
年収
500 万以上
発行する
500 万未満
発行しない
0 回
年収
300 万以上
発行する
300 万未満
発行しない

# 条件分岐 ー
# 場合分けを使った計算

```
def max(x,y)
  if y < x
    x
  else
    y
  end
end
```
max.rb

```
irb(main):003:0> load("./max.rb")
=> true
irb(main):004:0> max(123, 456)
=> 456
irb(main):005:0> max(max(12, 34), max(56, 78))
=> 78
```

# 3.1.2 3 通りの場合分け (p.34)

```
def sign(x)
    if x < 0
        -1
    else
        if 0 < x
            1          # not(x<0) and 0<x
        else
            0          # not(x<0) and not(0<x)
        end
    end
end
```

sign.rb

```
irb(main):002:0> sign(123)
=> 1
irb(main):003:0> sign(-3)
=> -1
irb(main):004:0> sign(0)
=> 0
```

# 3.1.3 複雑な条件 (p.35)

- 色々な比較

| 書き方 | 数学 | 意味 |
|---|---|---|
| x > y | $>$ | x が y より大きい |
| x >= y | $\geq$ | x が y 以上 |
| x == y | $=$ | x と y が等しい（x=y でないことに注意） |
| x < y | $<$ | x が y より小さい |
| x <= y | $\leq$ | x が y 以下 |
| x != y | $\neq$ | x と y が異なる |

- 条件式の組合せ

| 書き方 | 意味 |
|---|---|
| x > y \|\| x == 0 | x > y または x == 0 |
| x < y && y < z | x < y かつ y < z |
| !(x < y && y < z) | (x < y かつ y < z) でない |

# 3.1.3 複雑な条件 (p.35)

= でないことに注意
= は代入

- 色々な比較

| 書き方 | 数学 | 意 |
|---|---|---|
| x > y | $>$ | x が y より大きい |
| x >= y | $\geq$ | x が y 以上 |
| x == y | $=$ | x と y が等しい（x=y でないことに注意） |
| x < y | $<$ | x が y より小さい |
| x <= y | $\leq$ | x が y 以下 |
| x != y | $\neq$ | x と y が異なる |

- 条件式の組合せ

| 書き方 | 意味 |
|---|---|
| x > y \|\| x == 0 | x > y または x == 0 |
| x < y && y < z | x < y かつ y < z |
| !(x < y && y < z) | (x < y かつ y < z) でない |

# クイズ: 正しいのはどれ？

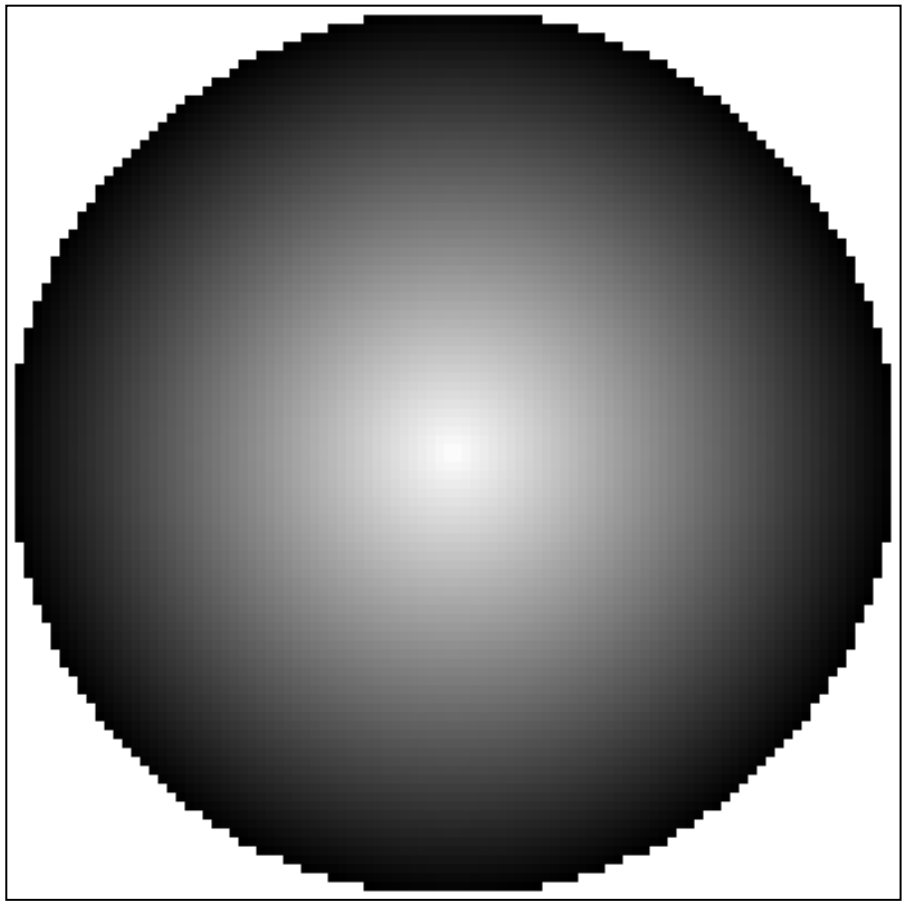
x の値が 7、y の値が5、z の値が 3であるとして

1. x < y
2. x <= y
3. x < y && y != z
4. x <= y || y == z
5. !(x < y && y == z)

| 書き方 | 意味 |
|---|---|
| x > y \|\| x == 0 | x > y または x == 0 |
| x < y && y < z | x < y かつ y < z |
| !(x < y && y < z) | (x < y かつ y < z)でない |

| 書き方 | 数学 | 意味 |
|---|---|---|
| x > y | $>$ | x が y より大きい |
| x >= y | $\geq$ | x が y 以上 |
| x == y | $=$ | x と y が等しい |
| x < y | $<$ | x が y より小さい |
| x <= y | $\leq$ | x が y 以下 |
| x != y | $\neq$ | x と y が異なる |

# 3.4 繰り返しによる画像の作成 (p.40)



- 座標値から明度を計算
  - 画像の中心からの距離
- 100x100の配列を作る
- 繰り返しによって
  各点の明度を与える
- 配列を表示

# 練習3.2

- 配布プログラム sphere.rb, make1d.rb, make2d.rb をダウンロード
- sphere.rbの中の関数bを式3.2に従って完成 (distance.rbの中にdistance(x,y,u,v)が定義されていること[p.10,練習1.2a])

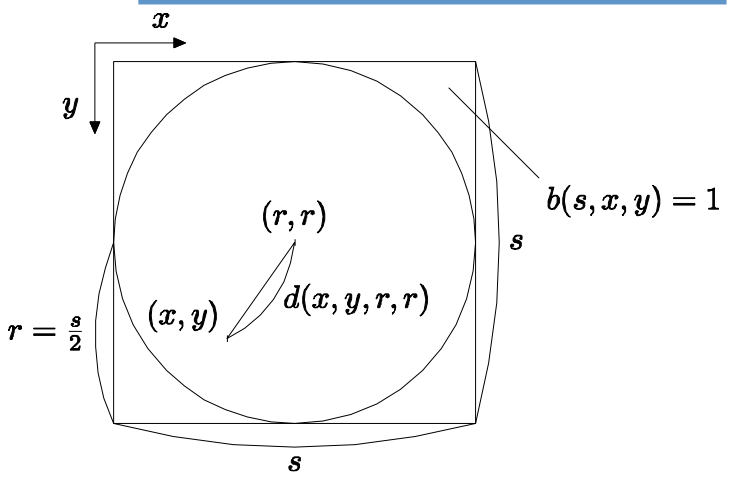
$$b(s,x,y)=\begin{cases}\dfrac{r-d}{r} & (d\le r)\\ 1 & (d>r)\end{cases}$$

$$\text{where } r=\frac{s}{2},\ d=\text{distance}(x,y,r,r)$$

完成したら「1」を投票し
show(sphere(100))を実行

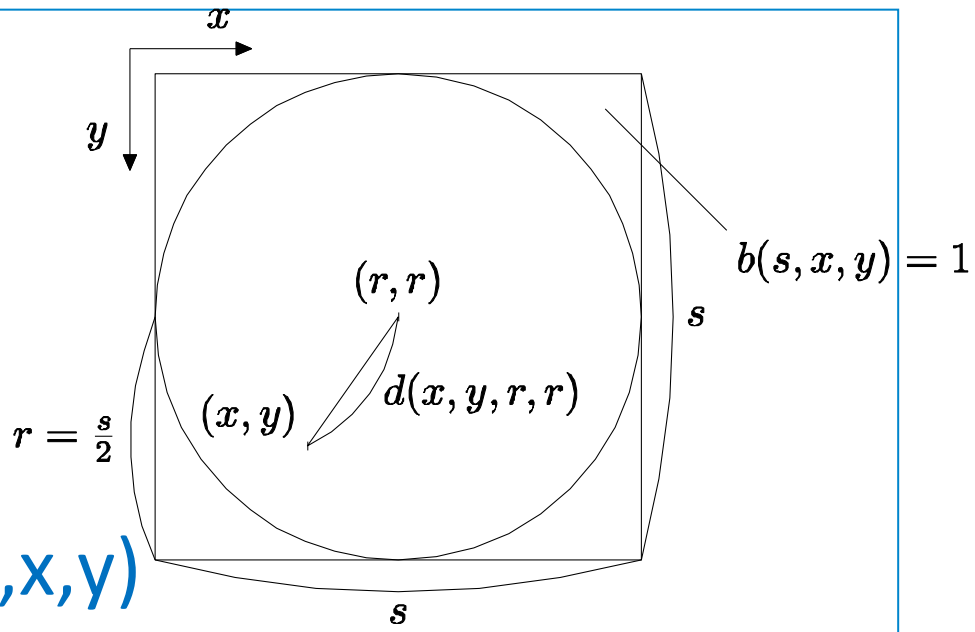
動作確認

```
> b(100,50,50)
1.0
> b(100,40,50)
0.8
> b(100,50,50)
1.0
> b(100,0,50)
0.0
> b(100,5,5)
1
> b(100,15,15)
0.010050506338833 4
> b(100,95,95)
1
> b(100,85,85)
0.010050506338833 4
```



# 2 重の繰り返し

```
def sphere(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s -1)
            image[y][x] = b(s,x,y)
        end
    end
    image
end
```




show(sphere(20)) を実行して表示される画像を確めよ。

sphere.rb